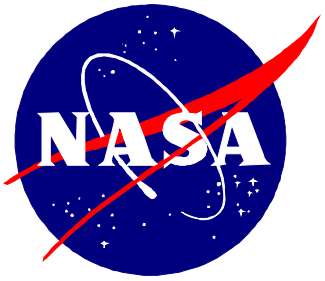
NASA Contractor Report 201701



PIV Data Validation Software Package

James L. Blackshire
ViGYAN, Inc., Hampton, Virginia

CONTRACT NAS1-19505

June 1997

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

PIV Data Validation Software Package

Abstract

A PIV data validation and post-processing software package was developed to provide semi-automated data validation and data reduction capabilities for Particle Image Velocimetry data sets. The software provides three primary capabilities including 1) removal of spurious vector data, 2) filtering, smoothing, and interpolating of PIV data, and 3) calculations of out-of-plane vorticity, ensemble statistics, and turbulence statistics information. The software runs on an IBM PC/AT host computer working either under Microsoft Windows v3.1 or Windows 95 operating systems.

Acknowledgments

The author wishes to thank several people for assistance and input to the software development. Foremost of all are Mr. William Humphreys and Mr. Scott Bartram, who provided technical support and guidance with PIV theory and practice. Their patience and helpfulness are greatly appreciated. The author also wishes to acknowledge Dr. C.S. Yao and Mr. Keith Pascal for help with the turbulence statistics and vorticity algorithms. Finally, I gratefully acknowledge the support provided by the Measurement Science and Technology Branch at NASA Langely Research Center, Hampton, VA 23681, under contract No. NAS1-19505.

Contents

PIV Data Validation Software Package

1.0 Introduction

The IBM PC/AT PIV data validation and post-processing algorithm provides the capability for 1) removal of spurious vector data, 2) filtering, smoothing, and interpolating of PIV data, and 3) calculations of out-of-plane vorticity, ensemble statistics, and turbulence statistics information. The algorithm is stand alone and requires no hardware control or manipulation. The software runs on an IBM PC/AT host computer working either under Microsoft Windows v3.1 or Windows 95 operating systems. The processing is semi-automated, requiring user input to a configuration file prior to program execution for input of various validation system parameters. The executable program file is named **VALIDATE.exe** with the configuration text file called**VALIDATE.cfg**.

2.0 Software Description

2.1 General Software Description

The software is modular in nature and applies various data validation and post-processing steps in a sequential manner. The order of program execution typically performs the following steps:

1) system initialization
2) read in configuration file parameters
3) read in PIV raw vector data file(s)
4) apply bandpass validator (if user requests)
5) apply local median validator (if user requests)
6) apply 3x3 median filter (if user requests)
7) apply image shift subtraction (if user requests)
8) calculate ensemble mean and ensemble standard deviation (if user requests)
9) calculate vorticity (if user requests)
10) calculate turbulence statistics (if user requests)
11) write validated data to file(s)

The user decides which of these validation step(s) will be applied to the data by means of the validate.cfg configuration file. If a certain step is turned on in the configuration file, then it is executed. If a certain step is<u>not</u> turned on in the configuration file, then it is skipped. The program runs in a 'batch' mode type operation in which files needing validation are read in and validated one after another. This requires a specific type of file format where the base filename is the same for all the files and the file extension increments from *.000 to *.max, where max is the number of files being validated.

The program outputs five types of files. Validated vector files are the primary output, and are validated and written out one after another as each file is validated. They are output with the output base filename provided by the user in the configuration file, and with the same file extension number read in from the input vector file. The second type of file output is the ensemble statistic file, which writes out the results of the ensemble mean and standard deviation calculations. The ensemble statistics filename takes the same output base filename provided by the user in the configuration file, and has '*.add' as its file extension. The third type of file output is the turbulence statistics file, which writes out the results of the 2nd order turbulence statistics calculations. The turbulence statistics filename takes the same output base filename provided by the user in the configuration file, and has '*.tur' as its file extension. The fourth type of file output is also related to the turbulence statistics calculations, and involves the evaluation of fluctuation velocity files. These files have the same output base filename provided by the user in the configuration file with an added 'f' at the end of the base filename (e.g. validated vector filename: 'output.000', vorticity filename: 'outputf.000'). The file extension again takes the same file extension number read in from the input vector file. The final type of file output is the out-of-plane vorticity file(s), which write out the results of vorticity calculations to file. A vorticity file in this case is output for every input vector file. The files have the same output base filename provided by the user in the configuration file with an added 'v' at the end of the base filename (e.g. validated vector filename: 'output.000', vorticity filename: 'outputv.000'). The file extension again takes the same file extension number read in from the input vector file.

2.2 <u>Detailed Software Description</u>

The program is made up of two 'c' source files, and three resource files. The program is of a Windows 'quickwin' type which allows it to be run in Windows 3.1 or Windows 95, but allows minimal user interaction. Interaction and control of various

program execution parameters is provided by a configuration file as described in the introduction. The user edits the configuration file before program execution, and when the program is run, the configuration parameters are read in from that file. The program then displays the parameters on the monitor in a quickwin window and begins validation with no further user input.

The source file VALIDATE.C is the primary vector validation program and will be detailed in this section. The source file VALWIN.C develops and displays the quickwin window and will not be detailed here. The user is referred to the commented listing provided. The three additional resource files will also not be detailed here.

The source file VALIDATE.C contains eleven modular functions that perform the various validation operations. The main() function begins and ends program execution and performs memory allocation, initialization, and control functionality. Its first task is to call the read_config() function which reads the configuration file parameters into global variables for use later in the program. The main() function then continues on and prints the configuration parameters in the quickwin windows, followed by data array memory allocation calls. A looping function is called next which begins the actual validation processing of individual raw data files. A function called onetenhund() converts the current file extension number from an integer to a string so that input and output file names can be developed for reading and writing purposes. Once the input and output filename strings have been concatenated the program prints the current file number being validated in the quickwin window and calls the function go().

The function go() groups all of the other function calls in one place and controls which ones are called or not called based on what the user input in the configuration file. The function first reads in the vector file data based on its format type (number of columns of data in the file) based on one of two default types. It then applies the various validation parameter file procedures provided by the user in the configuration file. The specific procedures (function calls) given in order of their calling sequence include:

2.2.1 **bandpass():**

The bandpass() function performs a bandpass filter operation on the data. It is the most subjective part of the validation program and requires user input for upper and lower bandpass values. It is intended to be used on grossly deviant vectors relative to the majority of vectors in the field. It was found necessary to include such a validation step before the local median validator step (the local median validator is meant to be used as

the primary validator) because large deviant vectors tended to skew excessively the validation parameters of the local median in some instances.

The bandpass() function treats the u and v velocity components separately, and checks to see if the u and v values are within the maximum and minimum values provided by the user in the configuration file. If either component is out-of-bounds, the second and third centroid peak values are checked for out-of-bound conditions. If the second peak is within bounds it replaces the first peak value (if the first was out). Similarly, if the third peak value was within bounds, and the second and first were out, then it replaces the first. Both the u and v components must be within bounds or both are considered out. If none of the three peaks are valid, then the position is zeroed out.

Included in the bandpass() function, and prior to the actual bandpass operator execution, an additional 'zerocheck' call is performed if the user indicated that in the configuration file. This processing sub-step performs a check to see whether the correlation peak was very nearly vertical or horizontal (conditions which exist because of field edges or flair edges of objects in the image field). If performed, this validation check zeroes out the position if the peak was nearly vertical or horizontal.

2.2.2 local_median():

The local_median() function is the primary validation feature of the program and was found to be superior to other automated validation routines such as local mean, global mean, and local divergence. It compares the center value to its nearest eight neighbors based on their local median. It does this comparison based on an evaluation of the local mean and local standard deviation levels of the eight neighbors. If the difference of the local median and center value is more than three standard deviations of the local neighborhood (8 neighbors), then it is considered invalid. The u and v velocity components are again treated separately, and if one is found to be invalid, then both are considered invalid. The second and third centroid values are again checked if the first is found to be invalid as described previously. If all three values are found to be invalid, the position is zeroed.

2.2.3 smooth():

The option of performing a 3x3 local mean filter function is then applied to the data. If the user indicated, the local mean is calculated based on the eight nearest neighbors of a point, and if there are two or more non-zero neighbors, the center point is

replaced with the local mean. This provides an interpolation if the center point was originally zero, and otherwise a smoothing operation. The unaltered original data array is used for all local mean calculations, and the new 3x3 filtered data is written to a new temporary array, so that corruption of the original data is not allowed.

2.2.4 unshift():

The capability of removing image shift data from the flow data files is provided by the unshift() function. If this option is chosen, the image shift filename provided by the user in the configuration file is used to read in the image shift data using the read_cal() function. If both the flow data and image shift data sets are non-zero for a given position, the image shift is subtracted from the data.

2.2.5 ensemble

The ensemble average routine keeps a running count of the validated u and v vector component values, summing each position for the incrementing validated files from *.000 to *.max. When all data files have been validated, and the running ensemble sum has been evaluated for each point in the flow, the ensemble sum at each point is divided by the number of valid sum values, which gives the ensemble mean at each point:

$$\text{for } u \neq 0 \text{ and } v \neq 0 \qquad \bar{u} = \frac{1}{\max} \sum_{i=0}^{\max} u_i \qquad \text{and} \qquad \bar{v} = \frac{1}{\max} \sum_{i=0}^{\max} v_i$$

where only non-zero values are included in the ensemble calculation for each point in the flow.

2.2.6 standard_dev():

Following the ensemble mean calculation, an ensemble standard deviation routine is run. The ensemble standard deviation keeps a running count of the square of the difference between the ensemble mean and the validated data point:

$$\text{for } u \neq 0 \text{ and } v \neq 0 \quad sdu = \sqrt{\frac{\sum_{i=1}^{\max} (u_i - \bar{u})^2}{\max - 1}} \qquad \text{and} \qquad sdv = \sqrt{\frac{\sum_{i=1}^{\max} (v_i - \bar{v})^2}{\max - 1}}$$

where again, only non-zero values are included in the ensemble calculation for each point in the flow.

2.2.7 vorticity():

The out-of-plane (spanwise) component of vorticity is given by:

$$\omega_z = \frac{1}{2}\left(\frac{\partial v}{\partial x} - \frac{\partial \mathrm{u}}{\partial \mathrm{y}}\right)$$

where (u,v) are the in-plane (x,y) velocity components. By invoking Stokes' theorem to relate the circulation per unit area around a point of interest, a numerical approximation can be implemented to the above equation. Defining the closed contour by the eight points surrounding the node at which the vorticity is to be evaluated we obtain:

$$\begin{aligned}\omega_z = \frac{\Delta l}{(2\Delta l)^2} \times \Big\{ & u_{i,j-1} + \frac{1}{2}(u_{i+1,j-1} + v_{i+1,j-1}) \\ & + v_{i+1,j} - \frac{1}{2}(u_{i+1,j+1} + v_{i+1,j+1}) \\ & - u_{i,j+1} - \frac{1}{2}(u_{i-1,j+1} + v_{i-1,j+1}) \\ & - v_{i-1,j} + \frac{1}{2}(u_{i-1,j-1} + v_{i-1,j-1})\Big\}\end{aligned}$$

where Δl is the grid step size. The above equation was used to evaluate the local out-of-plane vorticity using the eight surrounding neighbors of a point in the flow for each individual data file. If any of the neighbors u or v components were zero, the vorticity was set to zero.

2.2.8 fluctuate():

The capability of removing the ensemble mean flow data from the individual flow data files is provided by the fluctuate() function. If this option is chosen, the fluctuating velocity subtraction filename (i.e. the ensemble mean filename) provided by the user in the configuration file is used to read in the ensemble mean flow data using the read_fluct() function. If both the individual flow data and ensemble mean flow data sets are non-zero for a given position, the mean flow is subtracted from the data.

2.2.9 tstat():

The turbulence statistics function tstat() calculates the 2nd order turbulence intensities <u'u'>, <v'v'>, and <u'v'>. Fluctuating velocity files are read in one at a time, where the fluctuating velocity components are multiplied and a running sum is calculated:

for $u \neq 0$ and $v \neq 0$

$$< u'u' >= \frac{1}{\max} \sum_{i=0}^{\max} (u_i - \bar{u})^2$$

$$< v'v' >= \frac{1}{\max} \sum_{i=0}^{\max} (v_i - \bar{v})^2$$

$$< u'v' >= \frac{1}{\max} \sum_{i=0}^{\max} \left[(u_i - \bar{u}) * (v_i - \bar{v}) \right]$$

where once again, only non-zero values are included in the turbulence intensity calculations for each point in the flow.

2.2.10 write_data():

Data is written out to files in two basic ways. Validated velocity, fluctuating velocity, and vorticity data is written out on a file-by-file basis, and each process results in an additional file for each data file that was input. The validated velocity data is written out by the function write_data(), while the fluctuating velocity, and vorticity data is written out to file in the body of the main() function. The ensemble statistic and turbulence statistic files result in a single file being written out for each. Each of these is also written out to file in the body of the main() function.

The output file formats for each process also differ and take one of four forms. The validated data and fluctuating velocity data files are output with the same format as the input raw data files and have eleven columns of data across. The data columns include the following data types:

column 1	column2	column3	column4	column5	column6	column7	column8	column9	column10	column11
x position	y position	1st u	1st v	1st peak	2nd u	2nd v	2nd peak	3rd u	3rd v	3rd peak

where u and v are the u and v velocity components, respectively, and peak is the relative centroid correlation peak height recorded during the analysis process. The 1st, 2nd, and 3rd

designations represent the strongest, 2nd strongest, and 3rd strongest correlation peak heights, respectively.

The vorticity file is output with only five columns of data across given by:

column 1	column2	column3	column4	column5
x position	y position	u	v	vorticity

where u and v are the validated velocity components, and vorticity is the calculated out-of-plane vorticity at the position x,y.

Ensemble statistics are output to file with eight columns of data across given by:

column 1	column2	column3	column4	column5	column6	column7	column8
x position	y position	mean u	mean v	sdu	sdv	# files	0

where mean u and mean v are the ensemble component means, sdu and sdv are the ensemble component standard deviations, and #files is the number of validated vectors available (used) for calculating the ensemble statistics for that point. The last column 8 is a dummy variable and is not used.

The final output data format for the turbulence statistic file has six columns of data across given by:

column 1	column2	column3	column4	column5	column6
x position	y position	<u'u'>	<v'v'>	<u'v'>	# files

where <u'u'> and <v'v'> are the turbulence intensity components, <u'v'> is the Reynold's stress , and # files is again the number of valid files used in calculating the turbulence statistics for that point.

3.0 Data Validation

3.1 Validation Procedures

The PIV data validation process is typically an iterative process in nature and requires the validation program to be run several times with different parameters and steps turned on each time. The basic validation process begins with editing of the validate.cfg file with a text editor program. This file takes the form shown below:

Software_Version_Number:	1.1
Apply_Bandpass_Validator_(y/n)?	y
Apply_Local_Median_Validator_(y/n)?	y
Apply_3x3_Local_Mean_Filter_(y/n)?	y
Apply_Image_Shift_Removal_(y/n)?	y
Apply_Vector_Zero_Checking_(y/n)?	y
Calculate_Ensemble_Average_File_(y/n)?	y
Calculate_Out_of_Plane_Vorticity_Files_(y/n)?	y
Calculate_Fluctuating_Velocity_Files_(y/n)?	n
Calculate_Turbulence_Statistics_File_(y/n)?	n
Upper_Bandpass_Threshold_Level_for_U:	-9.5
Lower_Bandpass_Threshold_Level_for_U:	-34.0
Upper_Bandpass_Threshold_Level_for_V:	50.0
Lower_Bandpass_Threshold_Level_for_V:	41.0
Number_of_Vectors_in_x_direction:	90
Number_of_Vectors_in_y_direction:	90
Input_Raw_File(s)_Base_Name:	f90r9.v4n
Output_Validated_Vector_File_Name:	f90r9.v4t
Input_Image_Shift_Removal_Filename:	r9add46.v6
Input_Fluctuating Velocity_Subtraction_Filename:	r9add46.add
Number_of_Files_to_Validate:	3

PIV validation configuration file: **VALIDATE.CFG**

Basically three types of input are required by the user. First, various processing operations are either turned on or off depending on the desired output. This is accomplished by editing the appropriate operation parameter with either a ‘y’ for yes, or ‘n’ for no. The second type of user input involves editing the seven numerical input values. Four are used for the bandpass operator function, and the remaining three are used for overall test condition inputs. In particular, two numbers are required for the number of vectors points in the x and y directions, respectively, and one number is required for the number of files included in the overall data set that need validated. The final type of user input requires various file input strings for either file basename inputs or full filenames.

Once the validation configuration file is edited and saved, the validation executable program is run. Depending on the type of processing to be done, and the number of files involved the processing should take several seconds to approximately 1-2 minutes. The program, when executing, displays the parameter set on the screen, and also displays a number representing the current data file being validated as it executes.

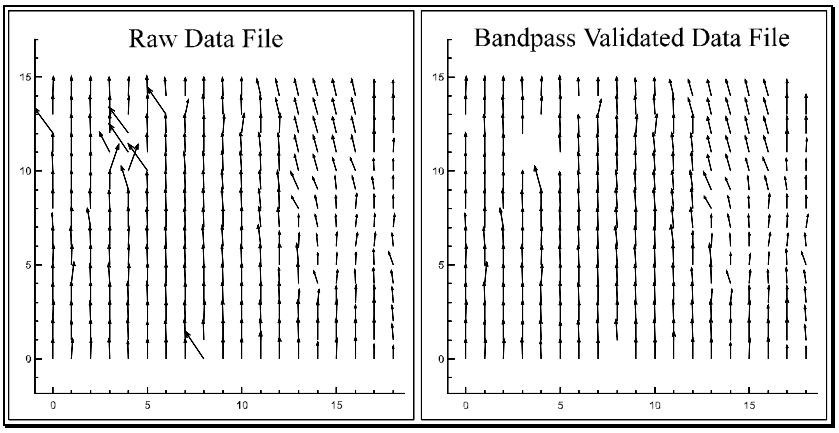
There are typically three types of validation and post-processing functions that the program will be used for, and each of these has a particular processing sequence that has in the past produced the desired results. They include:

1) Validation of Image Shift Data
2) Validation of Flow Data
3) Evaluation of Turbulence Statistics Information

Each of these are done in order, beginning with validation of image shift data. This is because the flow data usually needs a validated image shift subtraction file for validation to proceed. The turbulence statistics evaluation process likewise needs a validated ensemble mean flow file for processing to proceed. A brief description of the processing steps for each is provided below.

3.1.1 Image Shift Validation Steps

Image shift data validation requires three steps of processing. The first step involves determining appropriate u and v bandpass cutoff values for the bandpass validator algorithm. Raw PIV data typically includes a number of vectors that are dramatically different from the majority of the vectors in the data set. They are easy to distinguish visually, and their exclusion from the data set before further processing is done is the main reason for use of the bandpass validator operator. An example raw data file and bandpass validated file are depicted below for illustration.



The main objective of the initial bandpass operation is to remove obvious bad vectors, but also to be the least restrictive as possible. Choose upper and lower u and v component cutoff levels that just discriminate between the majority of good vectors and the few obvious bad vectors in the file. Also, when working with the validation of several

files at once, check to see that the upper and lower limits are suitable for all vector files being validated.

Once adequate upper and lower bandpass limits have been determined, the second step to image shift data validation involves including the new bandpass values in the configuration file. Also set the validator operator switches to **on** for:

1) the bandpass operator,
2) the local median validator,
3) the 3x3 local mean filter,
4) the vector zero checking,
5) and the ensemble averager,

and turn the remaining validator operator switches to **off**. Also input the raw data file basename for the analyzed data, the output validated basename you would like, the number of vectors in the x and y directions, and the number of files in the data set. Once this has all been input and saved to the configuration file, the validate.exe program is run.

Upon completion of the program, check the new ensemble average file that was created, and if necessary, make note of any vectors that appear to still be in error (record the x and y positions as well as the u and v vector component levels). If any are found, the third step involving manual extraction of bad vectors should be accomplished. This is done with a text editor, and involves opening the output ensemble average file with the editor, and inserting zeroes in place of the bad vector u and v component locations at the appropriate places. (The need for a manual vector extraction is sometimes necessary for the image shift ensemble file in that it will be used for subtraction for all of the flow data files in the next validation type, and if there are any obvious errors in the new image shift subtraction file they will propagate through all of the data to come).

Once the image shift ensemble average file has been completely validated, the processing of the raw flow data files can begin.

3.1.2 Flow Data Validation Steps

The validation of the flow data files follows a similar procedure to the image shift data validation and involves four steps. The first step again involves determining appropriate u and v bandpass cutoff values for an initial pass of the bandpass validator algorithm. The main objective of the initial bandpass operation is again to remove obvious bad vectors, but to be the least restrictive as possible. Choose upper and lower u and v

component cutoff levels that just discriminate between the majority of good vectors and the few obvious bad vectors in the file. And again, when working with the validation of several files at once, check to see that the upper and lower limits chosen are suitable for all vector files being validated.

The second step involves setting up the configuration file with the new upper and lower bandpass limits, and setting the validator operator switches to **on** for:

1) the bandpass operator,
2) the local median validator,
3) the image shift removal,
4) the vector zero checking,
5) and the ensemble averager,

and turn the remaining validator operator switches to **off**. Notice that the image shift removal function has been turned **on**, and the 3x3 mean filter has been turned **off**. Also input the raw data file basename for the analyzed data, the output validated basename you would like, the full filename of the image shift subtraction ensemble file, the number of vectors in the x and y directions, and finally the total number of files in the data set. Once all of this has been input to the configuration file, save it, and begin the validate.exe executable program.

The third step involves re-evaluating the bandpass validator upper and lower limits for the new data set that was just created with the image shift removed. A dual-pass approach of the bandpass and validator algorithms in general - once before image shift removal, and once after image shift removal, has proven over time to be the best approach for overall data validation and data rates. With the image shift removed at this point, flow features should be evident, and visual inspection of the flow and possible stray vectors should allow refinement of the bandpass limits. Again remember to check to see that the upper and lower limits check are suitable for all vector files being validated, especially if the flow is unsteady in nature.

The fourth and final step involves updating the configuration a final time with the new bandpass validator limits, and setting the validator operator switches to **on** for:

1) the bandpass operator,
2) the local median validator,
3) the 3x3 mean filter,
4) and the ensemble averager.

The calculate out-of-plane vorticity switch may also be optionally turned on at this point to create a set of output vorticity files. The input file basename should be set for the output file basename that was used in the previous step (actually step #2), and the output validated basename can again be set to what you would like. Once all of this has been input to the configuration file, save it, and begin the validate.exe executable program. This should produce a final validated flow data set and flow ensemble mean and standard deviation file. Manual extraction of stray vectors should not be needed, but after examination of the output files may be at your discretion.

Once the flow data's ensemble average file has been generated, the turbulence statistics processing can begin.

3.1.3 Turbulence Statistics Evaluation

The evaluation of turbulence statistics information requires two additional steps. A set of fluctuating velocity files must first be generated by subtracting the ensemble mean flow file from the individual validated flow data files. This is accomplished by setting the calculate fluctuating velocity switch to 'y'**yes** in the configuration file, and by setting the input file basename to the final validated flow data set's base filename, and the full filename of the ensemble average flow data filename in the fluctuating velocity ensemble subtraction filename space. Once this has been entered, and the configuration file has been saved, the validate.exe file should be executed. This will create a new series of files with the an 'f' appended to the beginning of the input base filename, and represents the fluctuating velocity components of the flow in each original data file.

The second step involves setting the calculate turbulence statistics switch to 'y'**yes** and all other switches to 'n' **no**. Set the input file basename to the new fluctuating velocity file basename created in step #1, and set the output base filename to the base filename you would like the turbulence statistics file to have. Once this is done and saved in the configuration file, execute the validate.exe one last time. This will create the turbulence statistics file with the base filename you input, and the *.tur extension.

4.0 Software Development Issues

The validation software was written in a Microsoft 'quickwin' format, which allows the program to be run in Windows 3.1 or Windows 95, but which doesn't allow any user interaction with the program while it is running. Five separate files are required during the compile and link process:

1) Validate.c - The validation program source file
2) Valwin.c - The quickwin source file
3) Validate.def - The definition file for the program
4) Validate.rc - The resource file for the program
5) Validate.ico - The bitmap icon for the program

In addition, a command line batch file has been used to control the compile, link, and resource compiler stages of the build process. This files name is Validate.bat. The program was compiled with Microsoft C compiler version v8.00, linker version v5.5, and resource compiler v3.11.

Editing of the code to add new functionality, or to change existing capabilities is relatively straight forward. The modular nature of the code allows this. The quickwin source file Valwin.c, definition file Validate.def, resource file Validate.rc, and icon file Validate.ico should not require any changes and should be left alone if possible.

A copy of the compile/link batch file listing is provided below:

```
cl -c -AL -Gsw -Zpe -DC_MSC -Dmain=Cmain validate.c valwin.c
link /NOD /NOE validate+valwin,,,libw Llibcew m4obM7WL pxipM7WL,validate.def
rc -K validate.rc validate.exe
```

It calls the Microsoft compiler, linker, and resource compiler, with various switches setup for a quickwin application.

This software package was developed under contract to NASA. Requests for copies of the source and executable codes described in this report should be directed in writing to one of the following:

NASA Langley Research Center
Technology Applications Group
Mail Stop 118
Hampton, Virginia 23681-0001

or

Vigyan, Inc.
Aeronautical Research Group
30 Research Drive
Hampton, Virginia 23666-1325